Modeling and Generating Complex Motion Blur for Real-time Tracking

Christopher Mei and Ian Reid
Department of Engineering Science
University of Oxford
{cmei,ian}@robots.ox.ac.uk

Abstract

This article addresses the problem of real-time visual tracking in presence of complex motion blur. Previous authors have observed that efficient tracking can be obtained by matching blurred images instead of applying the computationally expensive task of deblurring [11]. The study was however limited to translational blur. In this work, we analyse the problem of tracking in presence of spatially variant motion blur generated by a planar template. We detail how to model the blur formation and parallelise the blur generation, enabling a real-time GPU implementation. Through the estimation of the camera exposure time, we discuss how tracking initialisation can be improved. Our algorithm is tested on challenging real data with complex motion blur where simple models fail. The benefit of blur estimation is shown for structure and motion.

1. Introduction

Visual tracking is a fundamental step in many computer vision algorithms such as structure from motion and in robotic applications such as visual servoing. The tracking approach studied in this article belongs to the family of tracking methods that minimises a dissimilarity measure and more specifically a sum-of-squared-differences (SSD) between a reference template and the current image warped through a parametric transformation. This formulation leads to a non-linear optimisation problem that can be solved for small displacements (the type of movement that would be expected in a scene at video rate). Parametric tracking has been applied previously with a translational model [12], an affine model [15] or a more generic homography including illumination changes such as in [10]. A homography captures any motion induced by a planar region and so we also adopt this model. However our work is distinguished from [10] in that we also explicitly consider change in appearance induced by motion blur, the main thrust of our article.

Motion blur occurs in images captured by a camera when the camera or an observed object moves during the finite exposure time when the camera shutter remains open (or the CCD pixels remain exposed) to record the image. A handheld sequence with jitter or an indoor sequence with poor illumination are typical examples where motion blur has to be taken into account during tracking.

Recently Jin *et al.* [11] proposed an approach for tracking in presence of motion blur. The key observation was that instead of deblurring the image it is more efficient to match blurred versions of the templates (due to the commutativity of the blur kernels). The same idea is used in the work presented in this article. Our contribution is in the extension of their work from translational blur to *any* complex blur generated by constant velocity inter-frame motion between a camera and plane while *improving* the overall computational complexity. In particular, we relax the assumption of spatial invariance of the kernel enabling the modeling of complex blur such as rotation and “zooming”. We also detail how to generate motion blur efficiently and apply the algorithm to real-time tracking. Other contributions concern the minimisation process where we detail a novel hierarchy of transformations (applicable to other problems in computer vision) with gradual increase in complexity that improves the overall tracking stability.

In the first section, we discuss the formulation of the visual tracking problem. We detail the motion model, the motion blur model and how to minimise the cost function. For clarity, we did not include a robust cost function or basic photometric deformations as these could easily be added using for example the approach in [10]. Section 3 describes how to generate efficiently homography-based motion blur. Finally we show results obtained on simulated and real image sequences.

1.1. Related work

Several approaches can be considered to align blurred images: feature-based and direct approaches.

Extracting features in blurred images is a difficult problem. In the general case, a blur kernel can be infinitely complex and spatially variant. Some success has been achieved however under certain simplifying assumptions. In [9], the authors derive image moments invariant to blur and translation assuming the point spread function (PSF) is centrally symmetric. This work was later extended in [8] to cope for rotation and blur in an arbitrary number of dimensions. If the assumptions are violated, these methods can still *empirically* give usable results but the precision is often affected. This is where direct approaches prove useful.

Direct approaches that evaluate the blur parameters can be classified according to the following criteria: assumptions made on the kernel (arbitrary, only translation, spatially invariant, ...), if the method requires deblurring at each iteration step, if we assume only blur in one of the images and the type of application: deblurring two images without any relation between the blur and the inter-frame motion (for example digital images of the same scene taken at different times) or if we are tracking in a video stream.

A simple generic approach to registration would be to first apply blind deconvolution to the images using for example recent work such as [7]. However this would discard the extra information available when jointly estimating the different blur kernels and the motion and would also suffer from deconvolution artifacts.

The recent work by Yuan *et al.* [16] is well adapted to estimating the blur between two digital images taken with different exposures. The authors formulate the problem of registration in presence of blur as that of finding the sparsest kernel blur. The advantage of the approach is the possibility to evaluate arbitrary kernels. The flexibility of the approach comes at a high computational cost as each parameter of the kernel has to be estimated. The kernels are also assumed to be spatially invariant.

Our work is more closely related to methods where simplifying assumptions can be made on the blur kernel such as in video streams leading to more efficient algorithms applicable to real-time applications. In [14], the authors register blurred images by deblurring the input images assuming translational blur. The work by Jin *et al* [11] improves over the approach by using the commutativity of the blur kernels to track between *blurred* versions of the input images thus avoiding the expensive and ill-conditioned step of deblurring. [6] extends this work by adding a segmentation step and assuming different translational blurs for each region.

By restricting the blur kernel to a translation kernel, these approaches cannot deal with simple rotational blurs caused by camera rotation about the optical axis, or blurs induced either by a zooming camera or looming motion. Our work aims at removing these limitations by providing a way to model more complex blur and generate blurs efficiently on a GPU. Our main focus is improved tracking performance with applications in structure from motion from video sequences rather than the deblurring (ie image reconstruction) process itself. We can therefore readily make the assumption that a non-blurred version of the template is available from a previous observation.

2. Visual tracking

2.1. Motion model

Let $\mathcal{I}^*$ be the reference image. We will call reference template noted $\mathcal{R}$ a region $\mathcal{I}^*$ corresponding to the projection of a planar region of the scene. Let $\mathcal{I}_t, t = 1..T$ be a sequence of images. Tracking corresponds to finding the projection of the pixels $\mathbf{p} \in \mathcal{R}$ in the sequence of images.

The projection of a point belonging to a planar region follows a planar homography noted $\mathbf{H}$ defined up to a scale factor. Thus if the template has only undergone a geometric transformation, tracking at time t will correspond to finding the optimal homography $\overline{\mathbf{H}}$ such that:

$$\forall \mathbf{p} \in \mathcal{R},\ \ \mathcal{I}_t(\Pi(\overline{\mathbf{H}}\mathbf{p})) = \mathcal{I}^*(\Pi(\mathbf{p})) \tag{1}$$

with Π the standard perspective projection function $(\Pi(X, Y, Z) = (\frac{X}{Z}, \frac{Y}{Z}))$. For clarity, we will drop the projection function except during the Jacobian calculations.

Written as an optimisation problem, knowing an initial estimate $\widehat{\mathbf{H}}$, we wish to find the optimal increment $\mathbf{H}_0$ that minimises the sum-of-squared differences:

$$\mathbf{H}_0 = \mathrm{argmin}_{\mathbf{H}_i} \sum_{\mathbf{p} \in \mathcal{R}} \|\mathcal{I}_t(\widehat{\mathbf{H}}\mathbf{H}_i\mathbf{p}) - \mathcal{I}^*(\mathbf{p})\|^2 \tag{2}$$

In [2], the authors propose to fix the scale factor of $\mathbf{H}$ by choosing $\det(\mathbf{H}) = 1$ or equivalently by imposing $\mathbf{H} \in \mathbb{SL}(3)$ (Special Linear Group) through the local parameterisation of the increment using the exponential map:

$$\mathbf{H}_i = \mathbf{H}(\mathbf{x}) = \exp\left(\sum_{j=1}^{8} \mathbf{x}_j \mathbf{G}_j\right) \tag{3}$$

with $\mathbf{G}_1, ..., \mathbf{G}_8$, 8 generators of the Lie algebra $\mathfrak{sl}(3)$.

What was not noted and is of interest for computational efficiency and practical reasons is that affine transformations and translations can be written as connected Lie subgroups of $\mathbb{SL}(3)$. Affine transformations with 6 degrees of freedom can be written in two different ways:

$$\mathbf{A} = \begin{bmatrix} a_1\, a_2\, t_1 \\ a_3\, a_4\, t_2 \\ 0\ \ 0\ \ 1 \end{bmatrix} \text{ or : } \mathbf{A} = \begin{bmatrix} a_1\, a_2\, t_1 \\ a_3\, a_4\, t_2 \\ 0\ \ 0\ \ a_5 \end{bmatrix} \text{ s.t. } \det(\mathbf{A}) = 1 \tag{4}$$

The advantage of this second formulation is that we impose $\mathbf{A}$ to be invertible. Furthermore, $\mathbf{A}$ can be parameterised locally by the 6 first generators of the $\mathbb{SL}(3)$ Lie group and this automatically imposes the condition $\det(\mathbf{A}) = 1$. Similarly translations can be expressed using $\mathbf{G}_1$ and $\mathbf{G}_2$. This formulation has the practical advantage of generating a hierarchy of transformations similar to [3] but in a generic framework. The complexity of tracking is $O(m^2p)$ with m the number of parameters and p the number of pixels [1]. By adapting the number of parameters to the tracked template and motion, tracking efficiency and stability can be improved.

To summarise, with $\mathbf{H}(\mathbf{x}) \in \mathbb{SL}(3)$ representing a translation, affine transformation or homography according to the number of generators chosen, (2) can be written:

$$\mathbf{x}_0 = \text{argmin}_{\mathbf{x}} \sum_{\mathbf{p}\in\mathcal{R}} \|\mathcal{I}_t(\widehat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}) - \mathcal{I}^*(\mathbf{p})\|^2 \tag{5}$$

Let $\mathbf{p}{=}[x\ y\ 1]^\top$, $\mathbf{J}_{\mathbf{H}_\mathbf{x}}$ will be needed when minimising (5):

$$\mathbf{J}_{\mathbf{H}_\mathbf{x}} = \left.\frac{\partial \Pi(\mathbf{H}(\mathbf{x})\mathbf{p})}{\partial \mathbf{x}}\right|_{\mathbf{0}} = \begin{bmatrix} 1\,0\,y\,0 & x & -x & -x^2 & -xy \\ 0\,1\,0\,x & -y & -2y & -xy & y^2 \end{bmatrix} \tag{6}$$

With Matlab notations, the Jacobians for the translation and affine cases are $\mathbf{J}_{\mathbf{H}_\mathbf{x}}(:,1\!:\!2)$ and $\mathbf{J}_{\mathbf{H}_\mathbf{x}}(:,1\!:\!6)$ respectively. $\mathbf{J}_{\mathbf{H}_\mathbf{x}}$ is asymmetric and depends on the choice of generators.

2.2. Motion blur

In [11], the authors make the observation that the commutativity of the blur kernel enables tracking in presence of motion blur to be formulated as a minimisation problem over the *blurred* templates. This avoids the ill-conditioned and expensive operation of deblurring. We will follow the same idea but only assume that the current image is blurred. Therefore, we will be blurring the reference template for the tracking. If $\mathcal{I}$ is a blurred image, we denote its *un-blurred* version by $\mathcal{I}^u$.

In [11], a translational Gaussian kernel with parameter $\mathbf{v} = [v_x v_y]$ is chosen as a blur model:

$$\mathcal{I}_\mathbf{v}(\mathbf{p}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \mathrm{e}^{-\frac{t^2}{2}} \mathcal{I}^u(\mathbf{p} - \mathbf{v}t)dt \tag{7}$$

This choice of convolution is somewhat surprising as it does not correspond to the standard process of motion blur generation. In most cameras, the shutter speed can be considered as instantaneous so there is no Gaussian temporal blurring. We thus choose the more standard model, dubbed *directional* blur:

$$\mathcal{I}_\mathbf{v}(\mathbf{p}) = \int_0^1 \mathcal{I}^u(\mathbf{p} - \mathbf{v}t)dt \tag{8}$$

A more general model, valid for *any* blur generated by a planar template with constant velocity motion, can be written in homogeneous coordinates (with $\boldsymbol{\mathcal{H}}(\mathbf{x})$ the sum of the generators of $\mathbb{SL}(3)$):

$$\mathcal{I}_{\boldsymbol{\mathcal{H}}(\mathbf{x})}(\mathbf{p}) = \int_0^1 \mathcal{I}^u(\mathrm{e}^{-t\boldsymbol{\mathcal{H}}(\mathbf{x})}\mathbf{p})dt \tag{9}$$

It should be clear that this model encompasses equation (8). Let $\boldsymbol{\mathcal{T}}$ be the Lie representation of a translation. $\boldsymbol{\mathcal{T}}$ is nilpotent of index 2 so: $\mathrm{e}^{-t\boldsymbol{\mathcal{T}}}\mathbf{p} = (\mathbf{I} - t\boldsymbol{\mathcal{T}})\mathbf{p} = \mathbf{p} - \boldsymbol{\mathcal{T}}(:,3)t$.

Adding the motion blur model, with $(\overline{\mathbf{H}}, \boldsymbol{\mathcal{H}}(\overline{\mathbf{x}}))$ the values at the solution, we have the following identities:

$$\mathcal{I}_{t,\boldsymbol{\mathcal{H}}(\overline{\mathbf{x}})}(\mathbf{p}) {=} \int_0^1 \mathcal{I}_t^u(\mathrm{e}^{-t\boldsymbol{\mathcal{H}}(\overline{\mathbf{x}})}\mathbf{p})dt,\ \mathcal{I}_t^u(\overline{\mathbf{H}}\mathbf{p}) {=} \mathcal{I}^*(\mathbf{p}) \tag{10}$$

This leads to the relation:

$$\begin{array}{rcl} \mathcal{I}_t(\overline{\mathbf{H}}\mathbf{p}) & = & \int_0^1 \mathcal{I}_t^u\left(\overline{\mathbf{H}}(\overline{\mathbf{H}}^{-1}\mathrm{e}^{-t\boldsymbol{\mathcal{H}}(\overline{\mathbf{x}})}\overline{\mathbf{H}})\mathbf{p}\right)dt \\ & = & \int_0^1 \mathcal{I}^*(\overline{\mathbf{H}}^{-1}\mathrm{e}^{-t\boldsymbol{\mathcal{H}}(\overline{\mathbf{x}})}\overline{\mathbf{H}}\mathbf{p})dt \\ & \approx & \int_0^1 \mathcal{I}^*(\mathrm{e}^{-t\boldsymbol{\mathcal{H}}(\overline{\mathbf{y}})}\mathbf{p})dt \end{array} \tag{11}$$

In general $\overline{\mathbf{y}}$ is a function of $\overline{\mathbf{H}}$ and higher orders of t. This approximation is valid for small blur kernels. It proved efficient in practice. An alternative to avoid this approximation is to minimise in the current frame by warping the reference image ($\mathcal{I}_t^u(\mathbf{p}) = \mathcal{I}^*(\overline{\mathbf{H}}^{-1}\mathbf{p})$). In practice, we found that warping the current template lead to more stable tracking.

In [11], the authors claim that it is better to estimate the motion blur direction separately from the inter-frame motion. We will now detail the cost functions corresponding to the case of independent blur estimation and blur estimation linked to motion. The methods are compared in section 4.

We will use the following notation to incorporate prior knowledge $\widehat{\boldsymbol{\mathcal{H}}}$ of the blur parameters:

$$\mathcal{I}_{\widehat{\boldsymbol{\mathcal{H}}},\boldsymbol{\mathcal{H}}(\mathbf{x})}(\mathbf{p}) = \int_0^1 \mathcal{I}^u(\mathrm{e}^{-t\widehat{\boldsymbol{\mathcal{H}}}}\mathrm{e}^{-t\boldsymbol{\mathcal{H}}(\mathbf{x})}\mathbf{p})dt \tag{12}$$

Different blur directions (+8 unknowns)

$$\begin{pmatrix}\mathbf{x}_0\\ \mathbf{y}_0\end{pmatrix} = \min_{\mathbf{x},\mathbf{y}} \sum_{\mathbf{p}\in\mathcal{R}} \|\underbrace{\mathcal{I}_t(\widehat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}) - \mathcal{I}^*_{\widehat{\boldsymbol{\mathcal{H}}_b},\boldsymbol{\mathcal{H}}(\mathbf{y})}(\mathbf{p})}_{f(\mathbf{x},\mathbf{y})}\|^2 \tag{13}$$

$\widehat{\mathbf{H}}$ corresponds to the estimate of the motion obtained for the entire image sequence whereas $\widehat{\boldsymbol{\mathcal{H}}_b}$ is the current accumulated Lie algebra representation of the motion blur estimation for time t.

Blur linked to motion (+1 unknown) By assuming the motion and motion blur directions are linked, only the motion blur magnitude, λ, needs to be estimated:

$$\begin{pmatrix}\mathbf{x}_0\\ \lambda_0\end{pmatrix} {=} \min_{\mathbf{x},\lambda} \sum_{\mathbf{p}\in\mathcal{R}} \|\underbrace{\mathcal{I}_t(\widehat{\mathbf{H}}\mathbf{H}(\mathbf{x})\mathbf{p}) {-} \mathcal{I}^*_{(\widehat{\lambda}+\lambda)(\widehat{\boldsymbol{\mathcal{H}}_t},\boldsymbol{\mathcal{H}}(\mathbf{x}))}(\mathbf{p})}_{g(\mathbf{x},\lambda)}\|^2 \tag{14}$$

$\widehat{\mathbf{H}}$ corresponds to the estimate of the motion obtained for the entire image sequence whereas $\widehat{\boldsymbol{\mathcal{H}}_t}$ is the current accumulated Lie algebra motion estimation for time t and $\widehat{\lambda}$ is the current accumulated blur magnitude for time t. With e the exposure time, $e = \lambda f_r$ with f_r the frame rate.

2.3. Minimisation and Jacobians

The standard approach, called *forward-compositional* [12, 10], for minimising SSD cost functions is to use a first-order Taylor expansion around the identity transformations, ie use the current Jacobians. For (13) and (14), we obtain:

$$f(\mathbf{x}, \mathbf{y}) = f(\mathbf{0}, 0) + \left[\left.\tfrac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{0}} \; \left.\tfrac{\partial f}{\partial \mathbf{y}}\right|_{\mathbf{0}} \right] \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \qquad (15)$$

With $\mathcal{I}_t^W$ the current image warped through $\widehat{\mathbf{H}}$:

$$\left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{0}} = [\nabla_{\mathbf{q}} \mathcal{I}_t^W]_{\mathbf{p}} \mathbf{J}_{\mathbf{H}_{\mathbf{x}}} \qquad (16)$$

$$\left.\frac{\partial f}{\partial \mathbf{y}}\right|_{\mathbf{0}} = \left[\nabla_{\mathbf{q}} \int_0^1 t\mathcal{I}^*(e^{-t\boldsymbol{\mathcal{H}}_b}\mathbf{q})dt \right]_{\mathbf{p}} \mathbf{J}_{\mathbf{H}_{\mathbf{x}}} \qquad (17)$$

This last expression corresponds to a convolution weighted by time and can thus be calculated in the same way as the motion blur (Section 3).

$$g(\mathbf{x}, \lambda) = g(\mathbf{0}, 0) + \left[\left.\tfrac{\partial g}{\partial \mathbf{x}}\right|_{\mathbf{0}} \; \left.\tfrac{\partial g}{\partial \lambda}\right|_{0} \right] \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} \qquad (18)$$

$$\left.\frac{\partial g}{\partial \mathbf{x}}\right|_{\mathbf{0}} = \left([\nabla_{\mathbf{q}} \mathcal{I}_t^W]_{\mathbf{p}} + \widehat{\lambda} \left[\nabla_{\mathbf{q}} \int_0^1 t\mathcal{I}^*(e^{-t\widehat{\lambda}\boldsymbol{\mathcal{H}}_t}\mathbf{q})dt \right]_{\mathbf{p}} \right) \mathbf{J}_{\mathbf{H}_{\mathbf{x}}} \qquad (19)$$

$$\left.\frac{\partial g}{\partial \lambda}\right|_{0} = \left[\nabla_{\mathbf{q}} \int_0^1 t\mathcal{I}^*(e^{-t\widehat{\lambda}\boldsymbol{\mathcal{H}}_t}\mathbf{q})dt \right]_{\mathbf{p}} \mathbf{J}_{\Pi} \boldsymbol{\mathcal{H}}_t \mathbf{p} \qquad (20)$$

The Jacobian calculations are different from those in [11]; we first apply a convolution to the reference image and then differentiate instead of convolving the image Jacobian. This subtle difference makes it possible to write a generic Jacobian (valid for any sub-group of $\mathbb{SL}(3)$) without having to compute the Jacobian of the exponential.

From here we obtain the following first-order local minimisers (with $^+$ indicating the pseudo-inverse):

$$\left[\; \widehat{\mathbf{x}} \quad \widehat{\mathbf{y}} \; \right]^\top = -\left([\nabla_{\mathbf{x},\mathbf{y}} f]_{\mathbf{x}=\mathbf{0},\mathbf{y}=\mathbf{0}} \right)^+ f(\mathbf{0}) \qquad (21)$$

$$\left[\; \widehat{\mathbf{x}} \quad \widehat{\lambda} \; \right]^\top = -\left([\nabla_{\mathbf{x},\lambda} g]_{\mathbf{x}=\mathbf{0},\lambda=0} \right)^+ g(\mathbf{0}) \qquad (22)$$

In practice, this first-order approximation is improved by iterating the algorithm with further expansions around the previous solution.

2.4. Tracking initialisation

SSD-based tracking is well-adapted for obtaining precise sub-pixel estimates but its region of convergence rarely exceeds image displacements of more than a few pixels. At video-rate this is often sufficient except when large inter-frame motion occurs. To cope for these cases, SSD-methods are often initialised with feature matching approaches or by correlation search. Feature matching is difficult in presence of blur and correlation is also sensitive to blurring. By estimating the blur parameters, correlation scores can be improved by simply blurring the template during the search.

3. Motion blur generation

Generating motion blur has been extensively studied in particular in the field of computer graphics [13, 4]:

- Fast Fourier transform (FFT): this approach was used in [11, 16] and is only valid for spatially invariant kernels. The blur convolution becomes a product in the Fourier domain. The complexity is $O(\log(N)N)$ for N pixels and is independent of the blur magnitude,
- sampling by warping: the image is warped for different values of $\mathrm{e}^{t\lambda\mathbf{G}}$ and the values are accumulated to generate the desired blur. This "naive" method is easy to implement and parallelise. However it is not very efficient as the sampling rate should match the *longest* line integral to avoid aliasing effects which results in a complexity of $O(k_{max}N)$, with k_{max} the length of the longest line integral,
- line integral convolution [5]. This approach is used to visualise vector fields in physics. A simplified version has been used successfully in [4]. This approach is however computationally expensive and does not make use of the ease of evaluation of the vector field in different regions of the image.

Our approach for generating motion blur is similar to line integral convolution but we make explicit use of the structure of the problem to enable efficient calculation and parallelisation. The proposed algorithm has a complexity in $O(\hat{k}N)$, with $\hat{k}$ the average line integral length.

3.1. Homography-based blur

The proposed approach for the fast generation of homography-based blur is based on the recursive division of the integration domain. Let i_t be an integration step and $\mathbf{p}_k$ the coordinates of a pixel in the image; with:

$$\begin{cases} \mathbf{p}_0 = \mathbf{p} \\ \mathbf{p}_k = e^{-i_t\boldsymbol{\mathcal{H}}}\mathbf{p}_{k-1} \end{cases} \qquad (23)$$

the following recursive formulation enables the calculation of the curvilinear integral (trapezium or trapezoid rule):

$$\int_0^1 \mathcal{I}^*(\mathrm{e}^{-t\boldsymbol{\mathcal{H}}}\mathbf{p})dt = \sum_{k=0}^{\frac{1}{i_t}-1} \int_{ki_t}^{(k+1)i_t} \mathcal{I}^*(\mathrm{e}^{-t\boldsymbol{\mathcal{H}}}\mathbf{p})dt \quad (24)$$

$$= \sum_{k=0}^{\frac{1}{i_t}-1} \int_0^{i_t} \mathcal{I}^*(\mathrm{e}^{-t\boldsymbol{\mathcal{H}}}\mathbf{p}_k)dt \quad (25)$$

$$\approx (\textstyle\sum i_t) \displaystyle\sum_{k=0}^{\frac{1}{i_t}-1} \frac{\mathcal{I}^*(\mathbf{p}_k) + \mathcal{I}^*(\mathbf{p}_{k+1})}{2} \quad (26)$$

In practice, we can account for occlusion or pixels outside the image by only taking a partial sum, in this case $\sum i_t \neq 1$. This approach makes it possible to calculate the values sequentially. However the step value i_t is not known before the calculation and must be evaluated. The choice of i_t should ensure that each pixel on the streamline is visited to take into account high frequencies. For efficiency, we also wish to avoid numerous evaluations of the matrix exponential. To enable the precalculation of $\mathrm{e}^{-i_t\boldsymbol{\mathcal{H}}}$, we can choose i_t as a power of 2, $i_t = \frac{1}{2^{n_t}}$. An i_t with the required properties can be found in the following way:

$$\|\mathbf{p}_1 - \mathbf{p}_0\| < 1 \;\Leftrightarrow\; \|\mathrm{e}^{-\frac{1}{2^{n_t}}\boldsymbol{\mathcal{H}}}\mathbf{p}_0 - \mathbf{p}_0\| < 1 \quad (27)$$

and when $i_t \to 0$, (27) becomes:

$$\|(\mathbf{I} - \frac{1}{2^{n_t}}\boldsymbol{\mathcal{H}})\mathbf{p}_0 - \mathbf{p}_0\| < 1 \Leftarrow n_t = \lceil \log_2(\|\boldsymbol{\mathcal{H}}\mathbf{p}_0\|) \rceil \quad (28)$$

During the iterative calculation of the integral, n_t can be modified to take into account the possible change in curvature along the streamline.

Figure 1 shows a streamline generated by the proposed approach, we can see that each pixel along the streamline is visited by the algorithm. Figure 2 shows an example of a generated blur.

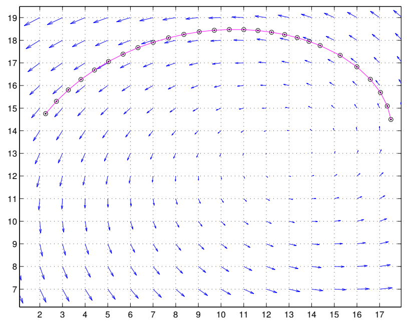
Figure 1. Example of a streamline with visited pixels

3.2. Parallelising the motion blur algorithm

The possibility of parallelising computer vision algorithms is becoming increasingly important as multi-core processors and easily programmable graphics processor units (GPUs) become common in desktop computers.

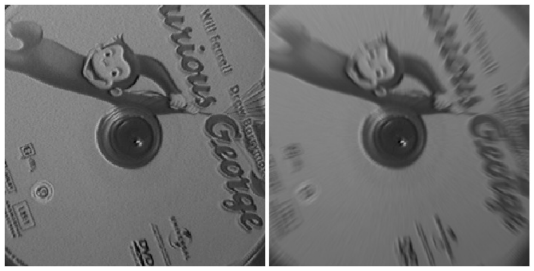
Figure 2. Example of a “zoom in” type of blur applied to an image

The proposed algorithm is readily parallelisable. We could imagine allocating a thread per streamline. However efficient parallelisation is a bit more subtle. SIMD (Single Instruction, Multiple Data) requires the computational load to be similar for the threads belonging to a same SIMD unit to avoid lost cycles. In the case of generating a rotational motion blur for example, if a thread A was allocated to the center of the rotation and a thread B furthest to the center and if A and B belonged to the same SIMD unit, A would have to wait for B to finish and this would require the same computational time as warping the entire image based on the longest streamline which is what we wish to avoid.

The key idea is that the *smoothness* of $\mathbb{SL}(3)$ means that locally the blur streamlines have similar lengths. Efficient blur generation can thus be obtained by simply dividing the image into local blocks, calculating the integral steps i_t for each block through equation (28) and starting threads belonging to the same SIMD unit on the same blocks.

The motivation for programming the current algorithm on a GPU is to show the possibility of video frame rate tracking (~30-100 Hz) using off the shelf hardware making tracking more robust with applications in structure from motion, visual servoing or augmented reality.

Figure 3 compares the time needed to generate a rotational blur of 10 deg. on a CPU (using one core at 2.40GHz), on a GPU using the naive approach and with the proposed method (GeForce 8800 GTX). There is a factor of 2 between the two GPU implementations. Blurring a patch of size 100×100 takes 12 ms, 0.89 ms and 0.44 ms respectively. For comparison, one iteration for a homography update takes 0.6 ms on the same CPU.

4. Experimental results

The following notations will be used to describe the different algorithms:

- E0 is a standard homography-based tracking algorithm without taking into account motion blur,
- E1 means only the blur magnitude is estimated (14),

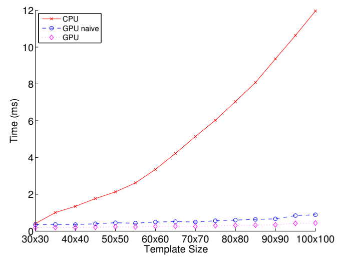

Figure 3. Timings for a rotational blur of 10 deg.

- E2 indicates translational motion blur estimation (2 parameters), *gauss* refers to the Gaussian model (7),
- E8 corresponds to the estimation of 8 independent blur parameters (13).

4.1. Simulated results

Simulations were made to evaluate the quality of the homography estimation. The simulated sequences are comprised of 20-30 images, the blur was generated using (9) with $\lambda = 0.7$. The sampling induced errors of 2-5 gray levels. The reprojection error (RMS) and the Frobenius norm with the respect to the true homography for a translation sequence (Fig. 4), rotation sequence (Fig. 5) and "zoom" sequence (Fig. 6) are shown. The pattern that emerges from these tests is that E8 gives low reprojection errors but is sensitive to noise and thus often leads to poor homography estimates. E2 is only adapted to translational blur or blur with a small magnitude. E1 gave reasonable results on all the tests. E2 *gauss* did not give satisfying results but this could be predicted as the simulation made use of a different blur model. When no motion blur model was used, the homography was generally poorly estimated.

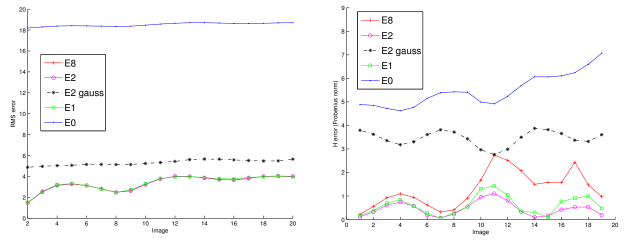

Figure 4. Sequence with translational blur

4.2. Real data

We use two sequences – one corrupted by a "standard" translational blur, and one of a rotating disc in which the blur is clearly not translationnally invariant in the image – in order to demonstrate our algorithm and evaluate it. Both example sequences were acquired in an indoor environment using a Unibrain Firewire camera with 640×480 resolution. The first sequence is a hand-held sequence with mainly translational blur. The second is a rotating DVD. The minimisation was done in a coarse-to-fine approach; the number of motion model parameters was adapted to the image texture at a given scale.

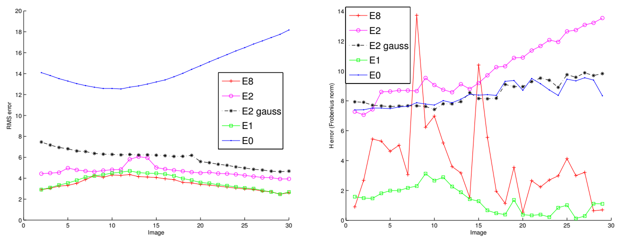

Figure 5. Sequence with rotational blur

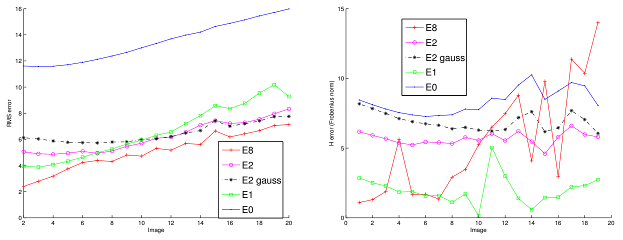

Figure 6. Sequence with "zoom in " blur

4.2.1 Hand-held sequence

Images in figure 7 show the visual tracking obtained on a hand-held sequence exhibiting mainly translational blur. The blurred reference image an the reprojected current image are also shown. Figure 8 details the RMS error during the image sequence. Images 41-42 and 52-57 are strongly blurred which explains the strong RMS errors for E0. In this sequence, the reprojection errors do not differ a lot between the different algorithms taking into account blurring. E1 is thus the best choice because the computational complexity is lower.

4.2.2 Sequence with rotational blur

Images of Fig. 9 show the visual tracking of a region on a DVD. The blurred reference images and the reprojection of the current image are also shown. Figure 10 is a plot of the RMS reprojection error during the sequence. The vertical black dotted lines indicate images where a strong amount of blur occurred.

The E1 algorithm showed higher errors than the algorithms with higher complexity *except* when motion blur occurred. The error was then lower than the E2 algorithms and comparable to E8. This is exactly the desired behaviour: these errors indicate that the blur is being correctly modeled without over-parameterisation. (We may note that lower er-

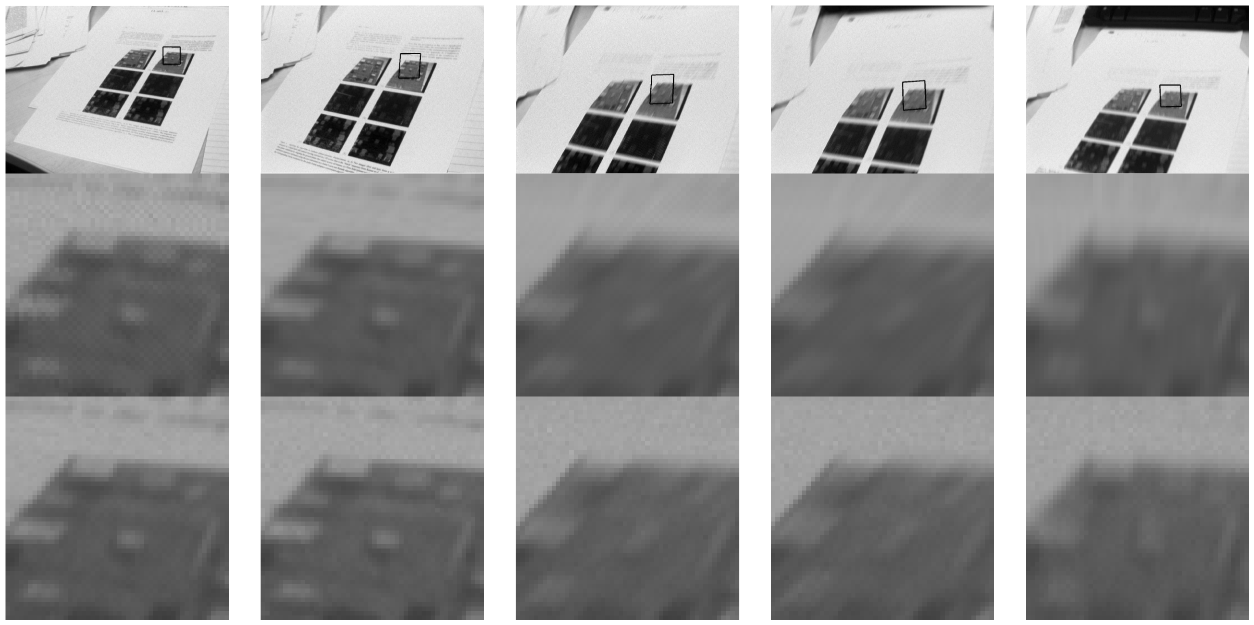

Figure 7. Images 1, 29, 42, 55 and 73 of the hand-held sequence with blurred reference image (middle row) and reprojected current template (bottom row) (algorithm E8)

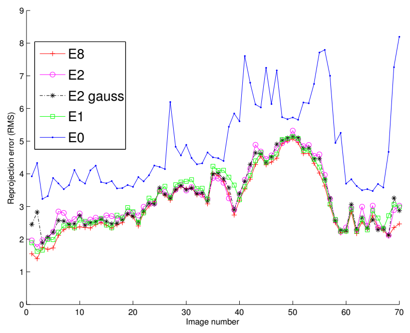

Figure 8. Reprojection error during the handheld sequence

rors for the blurred images come from the low-pass filter effect of blurring.)

5. Conclusion

In this article, we studied how to model and generate blur for improving visual tracking. Our approach improves over the current state of the art by enabling to track in presence of spatially variant blur such as rotation and zoom and more generally any blur generated by constant velocity motion between a camera and a planar template. Several formulations of the problem with different complexities were analysed and we came to the conclusion that for many real-world sequences a single blur magnitude parameter was sufficient. This result is interesting as it indicates that tracking can be greatly improved with a low increase in overall complexity and with good stability. A hierarchy of motion models was also proposed to improve tracking complexity.

References

[1] S. Baker, R. Patil, K. Cheung, and I. Matthews. Lucas-kanade 20 years on. Technical report, Robotics Institute, Carnegie Mellon University, 2004.

[2] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE International Conference on Intelligent Robots and Systems*, 2004.

[3] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, 1992.

[4] G. J. Brostow and I. Essa. Image-based motion blur for stop motion animation. In *Proc. of ACM SIGGRAPH'01*, 2001.

[5] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proc. of ACM SIGGRAPH'93*, pages

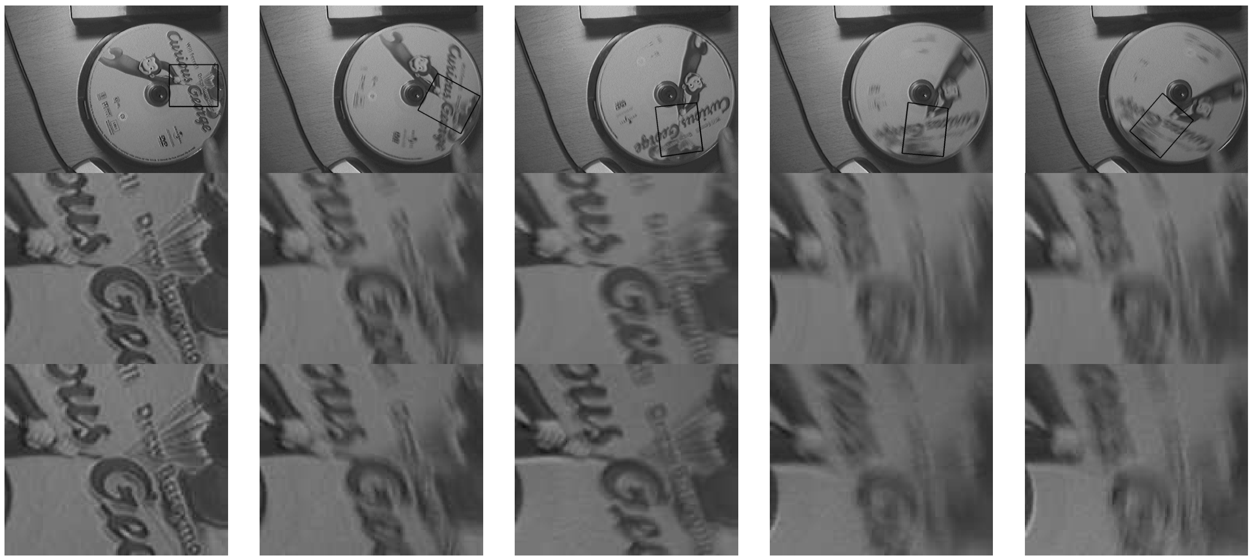

Figure 9. Images 301, 310, 333, 342 and 348 of the DVD sequence with blurred reference image (middle row) and reprojected current template (bottom row) (algorithm E1 dir)

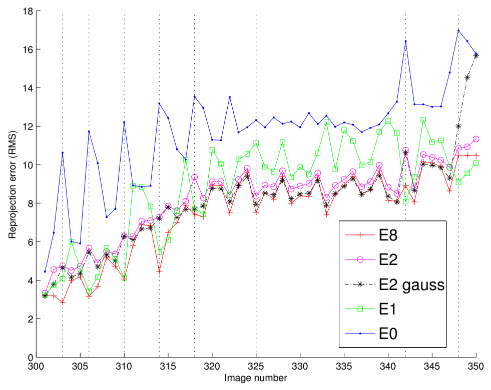

Figure 10. Reprojection error during the DVD sequence

263–270, 1993.

[6] S. Cho, Y. Matsushita, and S. Lee. Removing non-uniform motion blur from images. In *IEEE International Conference on Computer Vision*, 2007.

[7] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. 2006.

[8] J. Flusser, J. Boldys, and B. Zitová. Moment forms invariant to rotation and blur in arbitrary number of dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2), 2003.

[9] J. Flusser and T. Suk. Degraded image analysis: An invariant approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):590–603, 1998.

[10] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[11] H. Jin, P. Favaro, and R. Cipolla. Visual tracking in the presence of motion blur. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[13] M. Potmesil and I. Chakravarty. Modeling motion blur in computer-generated images. In *Proc. of ACM SIGGRAPH'83*, pages 389–399, 1983.

[14] A. Rav-Acha and S. Peleg. Two motion blurred images are better than one. *Pattern Recognition Letters*, 26(3):311–317, 2005.

[15] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[16] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. Blurred/no-blurred image alignment using kernel sparseness prior. In *IEEE International Conference on Computer Vision*, 2007.